Context-dependent determiners in logic programming: semantic representations and properties

Patrick Saint Dizier

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cédex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 532

CONTEXT - DEPENDENT DETERMINERS IN LOGIC PROGRAMMING: SEMANTIC REPRESENTATIONS AND PROPERTIES

Patrick SAINT-DIZIER

Juin 1986

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

CONTEXT-DEPENDENT DETERMINERS IN LOGIC PROGRAMMING: SEMANTIC REPRESENTATIONS AND PROPERTIES

Patrick SAINT-DIZIER
I.R.I.S.A.
Campus de Beaulieu
35042 RENNES-Cedex FRANCE

Publication Interne n° 297 - Mai 86
54 pages

ABSTRACT

This paper is devoted to the semantic representation of determiners such as "many", "few" and "little" using tools and formalisms of Logic Programming. For that purpose, the three branched quantified tree formalism is used as a basis and several extensions are proposed. Next, we study the persistence of the truth value of statements based on these determiners and interpreted either as facts, rules or integrity constraints. Finally, we show that several statements with "many" or "few" have rather to be represented by default rules than classical ones and we show how they can be generated.

Running Title: semantic representation of determiners.

RESUME :

Dans ce document, nous présentons un ensemble de représentations sémantiques pour des déterminants tels que "beaucoup de", "peu de", "la plupart", etc... Ces représentations dépendantes du contexte, sont définies à l'aide des outils et des formalismes de la programmation en logique. Ensuite, nous étudions la persistence de la valeur de vérité d'énoncés contenant ces déterminants et interprétés comme des faits, règles ou contraintes d'intégrité. Enfin, nous montrons que plusieurs classes d'énoncés doivent être représentées par des règles avec défaut plutôt que par des règles classiques et nous donnons une méthode pour construire ces défauts.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

Patrick SAINT DIZIER

CONTEXT – DEPENDENT DETERMINERS IN LOGIC PROGRAMMING : SEMANTIC REPRESENTATIONS AND PROPERTIES

Publication interne
n° 297

Mai 1986

SEMANTIC REPRESENTATIONS AND PROPERTIES

INTRODUCTION

Determiners such as "many", "most" or "few" have a meaning which is commonly admitted to be somewhat ambiguous, fuzzy and context-dependent. Consider the following sample dialogs:

Dialog 1:

- Do many french workers have a car ?

- Yes, 80% of them have.

- Does Michel have a car ?

- I have not the information, however, as Michel is a French worker, it is probable (80%) he has one.

Dialog 2:

- Has John many cars ?

- No, he has only 2 cars.

- And what about Kate ?

- Yes, she has 5 cars.

- Do people who have many cars pay an additional tax ?

- Yes, people who have more than 4 cars pay an additional tax.

Dialog 3:

- Did Chopin write many Mazurkas ?

- Yes, he wrote 51 Mazurkas.

- Did many other composers write Mazurkas ?

- No, to my knowledge, only Chopin has.

Dialog 4:

- Do many French artists live in poor conditions ? (with emphasis or focus on French)

- No, only few of them (20%) live with a salary under the average salary of artists in the world.

- And what about artists in general ? (with emphasis here on artists)

- Many of them (75%) live with a salary under the average salary of workers.

Dialog 1 introduces two problems: (a) the fuzziness of "many": in order to circumvent it, an appropriate answer seems to us to be the percentage of elements that satisfy the formula. In addition, the system answers "yes" if the percentage is greater than, for instance, 75%, which is, in this context, considered as the lower bound for "many". (b) default reasoning using a valid result as a default assumption.

Dialog 2 introduces the idea that "many" can have a relative meaning, e.g. Kate has more cars that the average number of cars people own. Notice that "many" is, at least, equivalent, in our example, to "more than 4".

Dialog 3 introduces the idea that "many", in some specific situations, cannot be represented using some average values and comparisons with the set of elements that do not satisfy a formula, since only Chopin wrote Mazurkas. With focus on "Mazurkas", notice that comparisons can be made on the number of pieces written in other styles by Chopin. However, it is necessary to be carefull in such comparisons because, for instance, it seems somewhat difficult to directly compare a Mazurka with an opera, the latter requiring much more work.

Finally, dialog 4 introduces the fact that, depending on location of focus in a sentence, quite different semantic representations and, thus, different answers are produced. Notice that adjectives (i.e. modifiers) as well as nouns may be subject to focus.

Our goal is to give a semantic representation to determiners, which cannot be represented using standard existential or universal quantifiers, in order to enable a system to produce answers of the kind of those given in the sample dialogs above. In this paper, we show that the logic programming approach is well adapted to deal with this problem, using , in addition to Horn Clause logic, some PROLOG calls such as setof, bagof or card and some tools defined in [27] such as AVERAGE. For related comparisons and a presentation of our general methodology, see [27].

In this work, we make a distinction between vagueness, ambiguity and context-dependence. By vagueness, we mean that determiners such as "several", "many" or "most" introduce a certain degree of fuzziness in the amount or quantity of entities they introduce. By ambiguity, we mean that a given determiner may be interpreted in different ways: "a" may be equivalent to "there exists", to "every" or to "all the". Finally, context-dependence means that some determiners may have different representations depending on context but also on the meaning of other elements in the sentence. Context-dependence is different from ambiguity since ambiguity at the level of determiners, from the point of view of a natural language parser-generator, is basically lexically induced whereas context-dependence is dealt with in semantic rules.

By lexically induced, we mean that the different possible representations of an ambiguous determiner are completely defined at the

level of the lexicon, without calls to the context to instantiate some parts of them. However, it is clear that calls to context are necessary to select the correct representation. Conversely, a context-dependent determiner is a determiner whose representation is partly uninstantiated in the lexicon but will become instantiated for precise sentences in given contexts from contextual information.

In this paper, we propose a framework to represent in a modular fashion vagueness, ambiguity and context-dependence. We will be here mainly concerned with context-dependence since results about fuzziness [36] and ambiguity [1,5,19] have already been developed within several frameworks and can be used here. We define semantic representations with a precision we think to be adequate for a man-machine dialog (cf. sections 2 and 4).

In section 3, we show how concepts can be generalized or associated to a set of equivalent ones using the IS-A hierarchy. Generalizations and equivalences are necessary when concepts are subject to focus.

Next, in section 5, we study the truth persistence of a representation. This problem, very akin to Truth Maintenance Systems [10], is of much interest when sentences are not questions but rather more general statements interpreted as rules or integrity constraints. Continuity between two states of a deductive database is defined and associated to results given in [16] to form an efficient system to control the persistence of the truth values of facts, rules and integrity constraints.

Section 6 is devoted to default reasoning. We concentrate on an automatic schema to produce default rules from sentences containing determiners such as "many" or "few". Finally, in section 7, we give the main lines of the implementation we have carried out.

1_ GENERAL FRAMEWORK

In this section, we introduce very briefly the main concepts and formalisms we use as a basis to our work. We emphasize on the three branched quantified tree formalism. Several extensions to this formalism for representing determiners, adjectives and some common adverbs have been proposed in [27].

First, it should be noticed that our ultimate goal is the specification of a friendly man-machine interface. This statement has at least two important consequences:

- we may assume that the domain of discourse and knowledge is objective and very limited, thus, it seems to us to be reasonable to assume that the number of facts, rules and integrity constraints is finite. In other terms, consequences of the closed world assumption are relevant here.
- We have to deal with linguistic and semantic phenomena with a degree of precision we think adequate for natural language front ends.

What we propose in the work is a set of tools whose formalism is based on that of Horn Clauses augmented by PROLOG predicates for which suitable proof procedures are available. These tools are, thus, directly executable in PROLOG. More details about our approach can be found in [27].

In our work, an input sentence is first parsed using any kind of logic-based grammar: from DCGs to Gapping Grammars [23,8,9]. The output is a three branched quantified tree [3,4,5,6]. It seems to us that this tree is a good intermediate representation because it reflects aspects of the syntactic and functional structure of the sentence which are relevant for the semantic stage. Furthermore, this tree can also be considered as the basic semantic representation of the sentence.

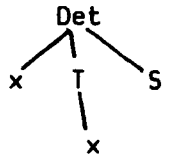
The second stage of the treatment is the computation of the semantic representation of the sentence from this three branched quantified tree. The general technique is contextual rewriting. Each subtree of the original three branched quantified tree is rewritten into a more precise representation. The rewrite systems we use are non-deterministic. Overgeneration is circumvented by adjunction of contextual preconditions to rules. Thus, a rule may be applied to compute the semantic representation of a sentence only if its preconditions are met. One of our goals is to find domain-independent preconditions [25]. This will be illustrated for determiners in the next sections.

The final stage is an evaluation process and, eventually, followed by an updating of the knowledge base. The knowledge base has the form of a deductive database [13,15,16,17]. Deductive databases has recently received considerable interest because of their many attractive and desirable properties. Such systems have first order logic as their theoretical basis which can be used for data, programs, queries and integrity contraints. Deductive databases have been proven to be more general than relational databases [17] and, furthermore, they can be implemented as efficiently than relational databases. They are also well adapted to knowledge representation and several systems use this approach. Finally, an interesting feature for our present purpose is the use of a PROLOG system as a query evaluator [6,13], even if the actual results require some restrictions on the form of formulas which can be used in the database.

We now introduce the three branched quantified tree formalism defined by [3,4,5,6]. Very briefly, in this representation, referential words (nouns, adjectives and verbs) are translated into predicates whose arguments are

typed. Determiners introduce a kind of meta-predicate whose arguments are predicates of the form:

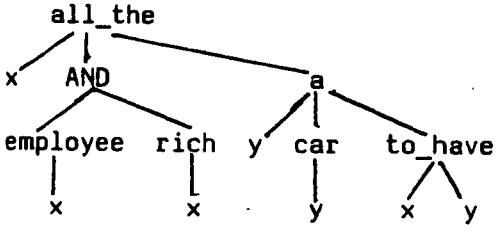
Det(x,T(x),S) or:



where Det is a determiner, T is the NP's translation introduced by Det (or a part of it) and S is the remainder of the sentence which is translated itself in the same manner. The subject NP is first translated and, thus, appears at the top of the tree, then complements in their input reading order and the verb are translated. Notice that Restr can include the representation of adjective phrases, noun complements and relative clauses. Thus, the sentence:

"All the rich employees have a car. "

is represented by:



This representation can directly be transformed into the PROLOG subset of first-order logic and thus, it is directly evaluable.

Notice that we do not use the rules proposed in [3,4,5] to deal with quantifier scoping. We adopt those proposed in [25].

We consider here that the three branched quantified tree representation is the result of the parsing process. We show in the next sections how nodes

labelled by determiners such as "many" or "almost" are rewritten into a more precise representation that can be a single predicate or a whole subtree.

However, the three branched quantified tree in its original form appears not to be enough informative for our purpose. Indeed, it is often necessary to have information about the syntactic category and the grammatical function of the predicates that represent adjective, nouns, verbs, etc... This will make the rewriting process much easier, less deterministic and, thus, more efficient. Futhermore, knowing the grammatical function is necessary to rewrite some determiners (cf. section 3) and also, for instance, to solve ambiguities and pronominal references. We will also add syntactic information such as gender, number and tense. A tree structure can be used for representing these informations in a concise way, as presented in [26].

Traditionnally, the logical representation of a sentence is embedded in a more general structure that include phonological, syntactic and semantic data, as in f-structures in Lexical Functional Grammars. In our appraoch, it seems more relevant to include some syntactic information as a specific branch in terminal symbols of the three branched quantified tree since (1) syntactic considerations are quite minor and (2) the rewriting process is more clear and performed more efficiently. This entails that the three branched quantified tree system is only an intermediate representation, with a certain explanatory power, but, in no case, it should be considered as a logical representation in its own.

To each nominal, adjectival and verbal predicate we add as first argument a term of the form:

<syntactic category>(gr(<grammatical function of the head for nouns>, <status>), feat(<gender>,<number>,<tense>,...))

where:
<syntactic category> is noun, adj., verb.
<grammatical function of the head for nouns> is subj, obj1, obj2, pp.
<status> is either head or modifier.

For instance, "rich" in the above example has the following additional argument:

adj(gr(subj,mod),feat(_,sing,_)).

This argument no longer appears in the final logical form. For more clarity, we shall not write this argument in the next sections but its use is explained in section 7.

Finally, after rewriting, the three branched quantified tree can directly be transformed into either a PROLOG goal if the sentence is a query [6] or a PROLOG program if the sentence is in the affirmative voice. Efficient evaluation methods are proposed in [31,17]. The query evaluation process has been proven to be sound and, furthermore, complete for definite and hierarchical databases [16]. Functions are allowed to appear in rules, queries and integrity constraints, with the restriction that a function always produces a finite set of values in a finite time for each instantiations of its free variables, which is the case in our framework. The number of possible instantiations has also to be finite. Theoretical problems are still under study about this latter point for a fuller integration in databases, queries and integrity constraints of aggregate functions that we use in our work (e.g. cardinal, average, sum, maximum and minimum).

2_ ON REPRESENTING "MANY"

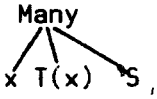
In a first step, we will focus on the representation of "many" because its representation subsumes the representation of the other context-dependent determiners. We use the same presentation than in [27] where the possible

representations of determiners are given under the form of rewriting rules. An important feature of this presentation is that it integrates both the description of the possible semantic representations for "many" and the way they are computed. Notice also that determiners are, a priori, rewritten independently of each other in any given sentence.

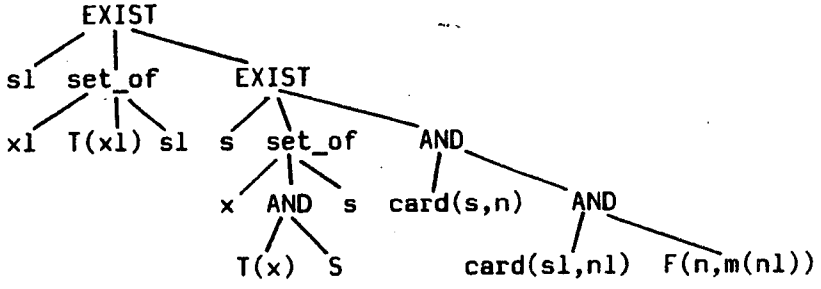
We define now seven representations for "many".

Representation I1 :

The idea behind this representation is to compare the set of elements that satisfy T(x)∧S(..x...) with those that satisfy T(x).

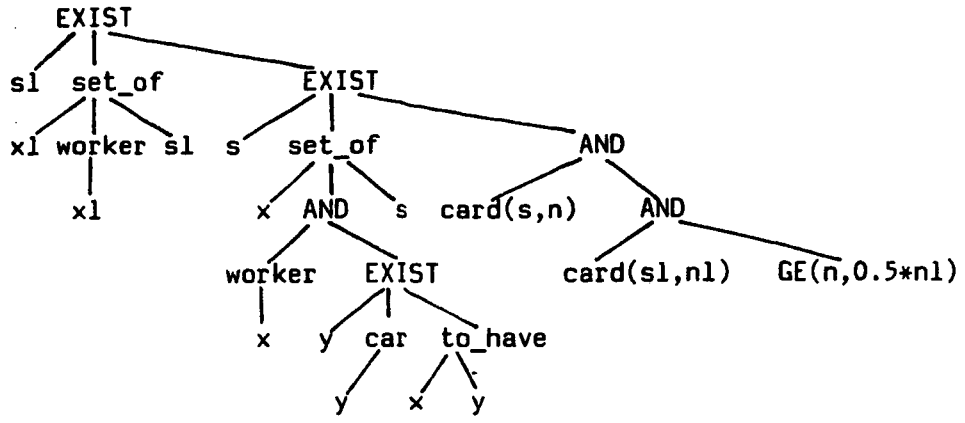


is rewritten into :



s1 represents the set of entities denoted by the noun that introduces Many whereas s represents the set of objects which satisfy the whole formula which is in the scope of "many".
F(n,m(n1)) is called the comparison predicate. F is a PROLOG call of the form: GE (greater or equal), LE (less or equal), EQ (equal), etc... and m is an arithmetical function. In our examples, m will often be very simplistic, however, notice that m may encode more complex results such as those of the fuzzy set theory. For "many" F is GE but it is represented here as a parameter because representation I1 can also be used for other determiners with F = LE or EQ (cf. section 4).

Example:
"Many workers have a car. "
is represented by:



We represent "many" by stating that the number of workers that have a car is greater than half the number of workers known to the knowledge base.

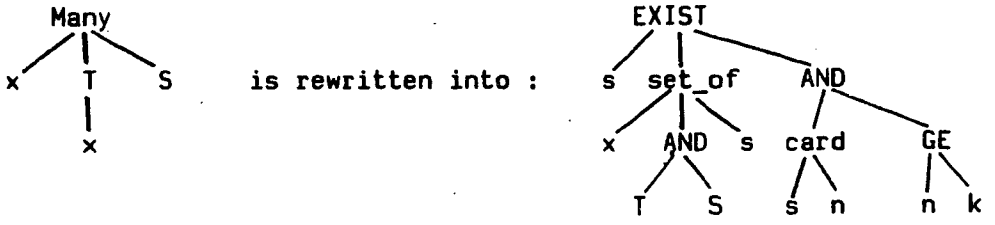
In fact, the comparison predicate can be used in two ways:

- to produce a positive or a negative answer, with m instantiated,
- to produce additional information (and thus a more cooperative response) if m is not instantiated in a first stage so as to allow the computation of n/n1, which directly gives a percentage. This approach is also more modular.

This enables the system to produce answers of the kind given in dialog 4 and in the first two sentences of dialog 1.

Representation I2 :

A very simple way of representing "many" is to state that it introduces a set of entities greater or equal to a given integer k, as in dialog 3. This integer may be fixed once for all or it may be a function of the size of the knowledge base KB: k = f(KB). Thus:



This representation is used when no relevant comparison criteria is available; it can also be viewed as a minimal condition guaranteeing that fewer than k will never be many. Thus, such a condition may also be added to the other representations.

Representation I3

The idea behind this representation is to compare the percentage P1 = n1/n2, where:

n1 = card(s1) , s1 = { x / T(x)∧S(..x..) is true}
n2 = card(s2) , s2 = { x / T(x) is true }

with the percentage P2 = n3/n4 where:

n3 = card(s3) , s3 = { x / S(..x..) is true}
n4 = card(s4) , s4 = { x / x is a semantically possible candidate for S that makes S true or false, depending on context }

To represent many(x,T(x),S) we state that P1 > P2.

For this to work, we have to define s4 more precisely. What we have to find, in fact, is a way to generalize T(x) in T'(x) where T' includes all the possible candidates semantically acceptable by S. Notice that this representation can be used only if S(x,...) is semantically relevant for T'(x). A simple way is to control the consistency of the semantic features of T' with those of the verb in S. In fact, it seems to us that the presence of focus implicitly entails this semantic consistency.

We now give an example of I3 and develop the generalization function in

section 3. For instance, consider the sentence:

"Many workers have a car. "

It may mean that:

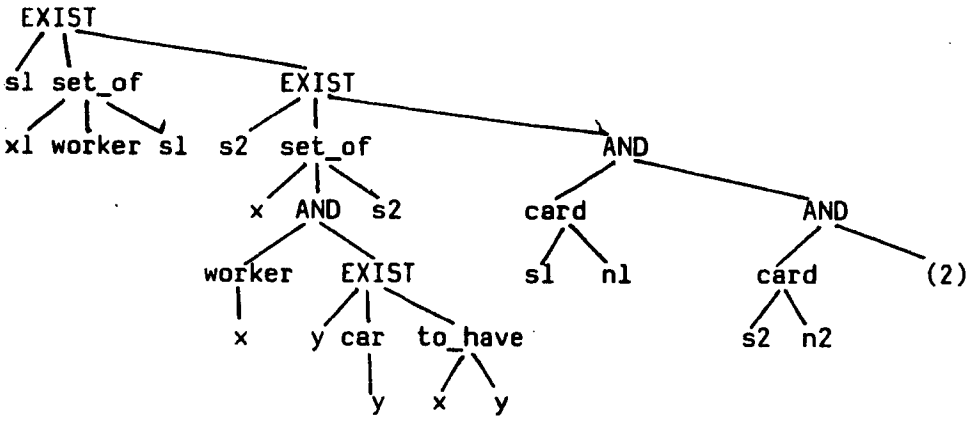
$$P1 = \frac{\text{cardinal of the set of workers that have a car}}{\text{cardinal of the set of workers}}$$

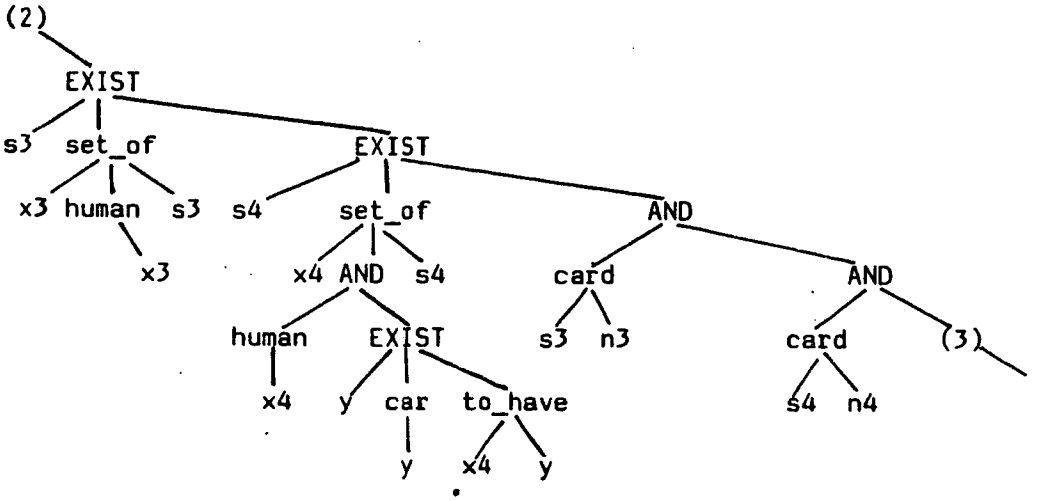
is greater than:

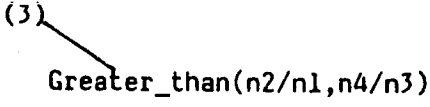
$$P2 = \frac{\text{cardinal of the set of humans that have a car}}{\text{cardinal of the set of humans}}$$

In this case, the generalization of worker(x) is human(x).

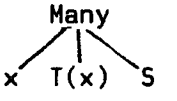
The final representation is :



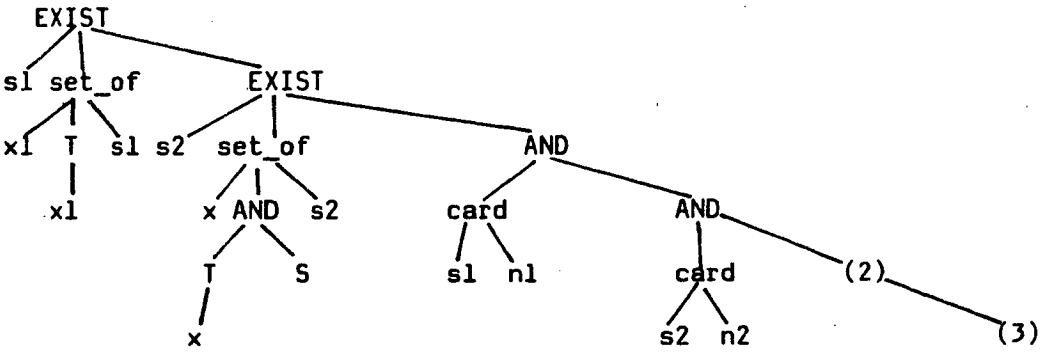




More formally :



is rewritten into a tree of the form:



(2) is identical to the tree above except that T is replaced by its generalization T' and s1, s2, x1 and x become respectively s3, s4, x3 and x4. (3) is: greater_than(n1/n2,n3/n4).

In this representation, we make the hypothesis that s2 and s4 are different from the empty set. We could also make this hypothesis for s2 in representation I1. This hypothesis may be viewed as a precondition to the selection of I3; it may also be viewed as a kind of presupposition.

Representation I4

The three previous representations assign to "many" what we call an absolute meaning. By absolute, we mean that the set of entities which is dealt with is considered as a whole. In our previous example, the set of workers is put into relation with the set of humans. We now introduce representations that assign to "many" a relative meaning. By relative, we mean that the set of entities we consider is, for the purpose of the sentence to represent, structured in the form of a set of sets. This phenomenon appears when "many" is directly dominated by a predicate that introduces a

relation between a set of entities and the set of entities quantified by many (cf. dialog 2). Futhermore, in this representation, entities introduced by "many" are not subject to focus. Relative meaning requires the use of the predicate AVERAGE that we have introduced in [27].

Consider the sentence:

"All the owners of many cars pay a tax. "

This sentence means that all people that own more cars than the average number of cars per people have to pay a tax. What we consider here is not the set of cars that people have as a whole, but the set of sets of cars per owner. The average number of elements per set of car can be computed by the predicate AVERAGE. A comparison predicate GE(n,m(val)), similar in its form to the one used in the previous representations is also necessary here.

For a measurable property prop possessed by some elements x of a set s (i.e., prop(x,xl) is known to the knowledge base), it is possible to compute the average value of prop for all the x, using the predicate AVERAGE, written as:

AVERAGE(prop,def,val)

where:

- prop is the name of the property (weight, height,...), (it is expressed by a predicate known to the knowledge base),
- def is the definition of the set, expressed either by the PROLOG call set_of(x,P,s) where P is a logical formula, or by an explicit list of elements.
- val is a real (or an "artificial" integer for it can be used by some PROLOG interpreters).

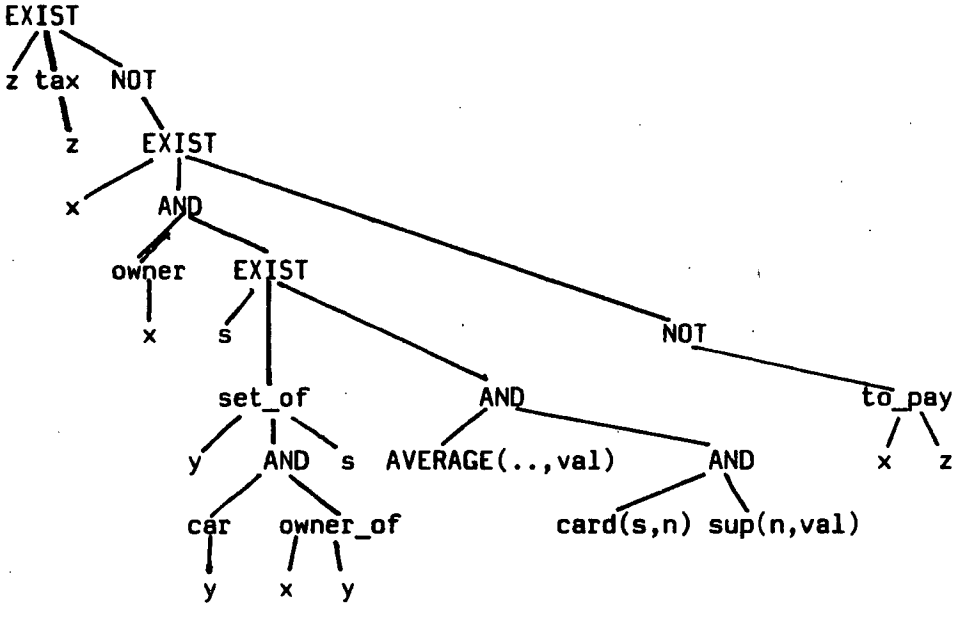
"val" is the average value of the xl computed from all the objects of the set s for which prop(x,xl) exists. When a formula contains a predicate AVERAGE, then AVERAGE is always true except if def is the empty set. An example of average value is the height of French men:

AVERAGE(height,set_of(x,AND(men(x),french(x)),s),val)

where val is instantiated to the average height of the French men.

For our present problem, the property prop is the cardinality of a set, that we note "card".

To represent the sentence above, it is necessary to compute the average number of cars per owner (or people, since we may include those that have no car). Then, we have to select those who have more than this average number of cars. The representation is the following:



where AVERAGE(...., val) is equal to :

AVERAGE(card,set_of(s1,EXIST(x1,owner(x1),set_of(y1,EXIST(y1,

car(y1),owner_of(x1,y1)),s1),s2),val)

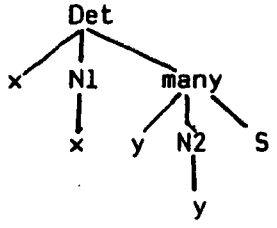
s2 is the set of sets s1, set of cars of a given owner x1. AVERAGE computes the average cardinal of the sets s1.

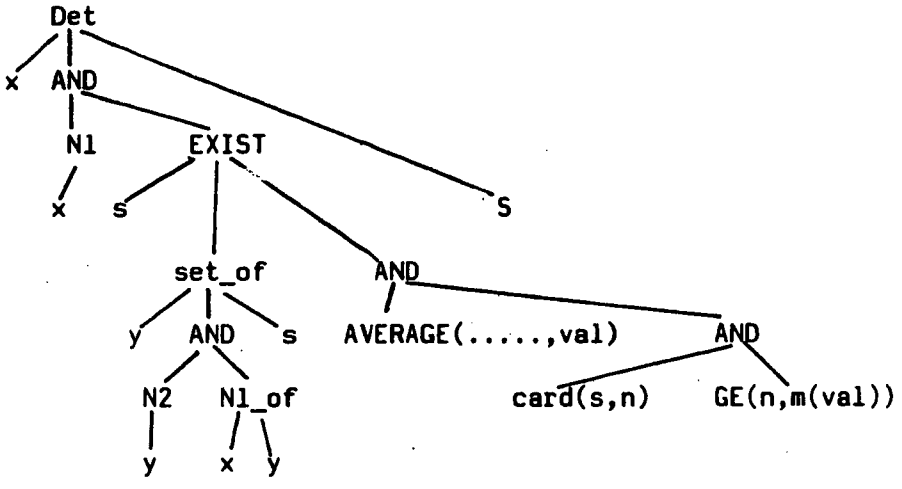
This representation is used for noun phrases where "many" is in the scope of another determiner. Let such a noun phrase be of the form:

det N1 of many N2

(N2 can also be included in a relative clause dominated by N1.)

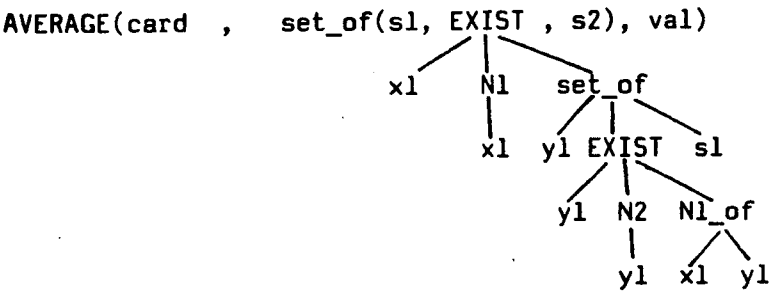
Then, the tree:





Notice that two instances of N1 are created: with one and two arguments. N1_of(x,y) can be transformed into a relative clause in a further stage, as explained in [27].

S contains the remainder of the sentence (the verb and, possibly, complements), AVERAGE is defined as follows:

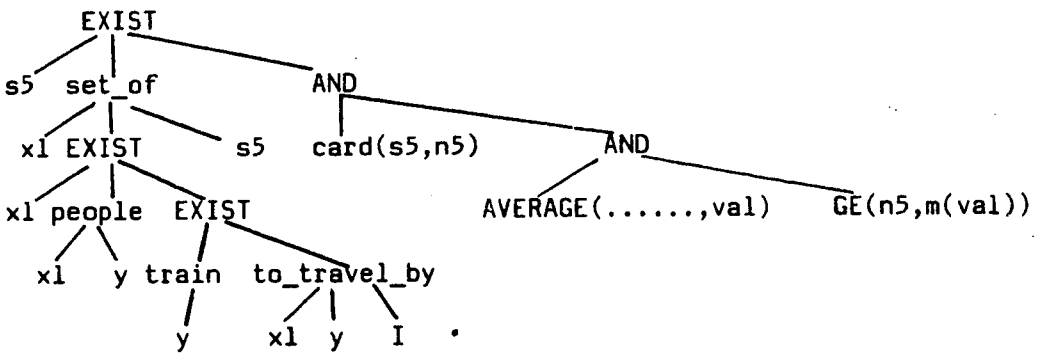


(Depending on the syntax of the PROLOG used, some minor syntactic modifications are necessary for the second argument of set_of).

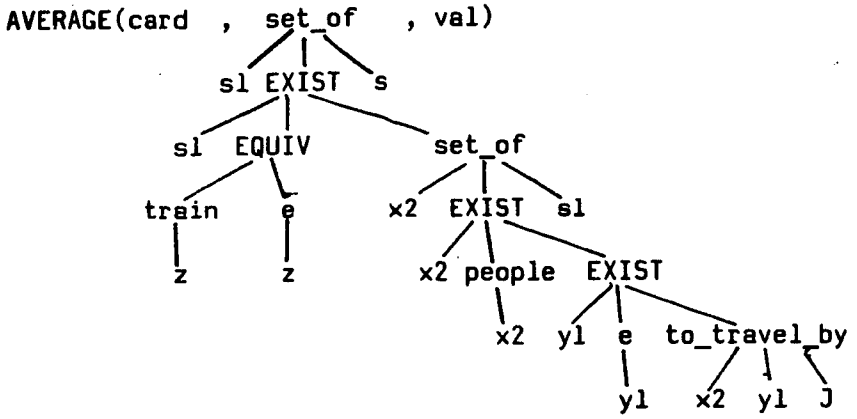
Representation I5

I5 allows one to represent sentences with absolute meaning, "many" in the subject noun-phrase and focus [12,18,19,20,21] on a noun-phrase complement. For instance, the sentence:

"Many people travel by train." with focus on "train" may mean that the number of people who travel by train is higher than the average number of people who travel by other means of travel known to the knowledge base (e.g. plane, train, bicycle, boat, ...). The predicate AVERAGE is used here to compute the average number of people who travel per mean of transport. Let val be this average number, then val is compared to n5, the cardinal of the set of people who travel by train in the following representation:



AVERAGE is defined as follows:



EQUIV is a predicate that delivers all the "sister"-concepts of train, i.e., for instance, boat, plane, car and train itself. EQUIV will be studied in detail in section 3. Note that EQUIV is composed of two arguments, the second argument (represented here by the variable e) contains a sister-concept of the first one.

Let the input form be: Many(x,T1(x),Det(y,T2(y),S(x,y,...))), where Det is any determiner. Then, the general form for I5 is deduced substituting "people" by T1, "train" by T2 and the verbal predicate by S(x,y,...). When there are several prepositional complements in the input sentence, they are decomposed into conjoined subtrees, in a way given in [27].

Representations I6 and I7

We end with two types of sentences with "many" introducing the head noun of a complement noun phrase N2, and with focus either on the subject N1 or on that complement. Consider, for instance, the sentences :

"A musician knows many musical works. "

"A musician can play many instruments. "

To represent the first sentence, EQUIV is used to deduce sister-concepts of musical work and in the latter, it is used to deduce other professions, which

are sister-concepts of "musician". To represent this type of sentence, it is necessary to consider both the noun-phrase N2 introduced by "many" and the subject noun phrase N1.

As for representation I3, these representations can be selected only if the sister concepts' semantic features for either the NP subject or complement are semantically consistent with those of the verb. Another condition is that there exists a mother node to the concept for which sister concepts are searched. Without mother node, sister concepts cannot be deduced, as for instance in:

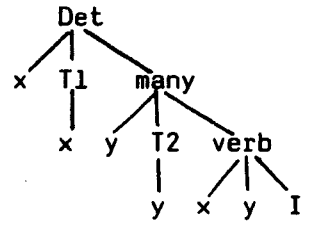
"Some birds fly over many countries each year. "

where country cannot have any equivalent concepts since:

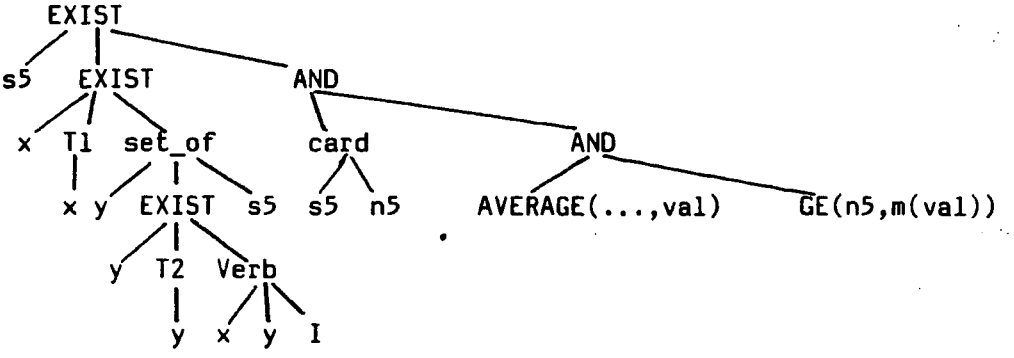
(1) either "country" has no mother node,

(2) assuming that a possible mother node is "object", then it is clear that "fly" cannot accept any sister node of "object" as a complement.

The subtree:



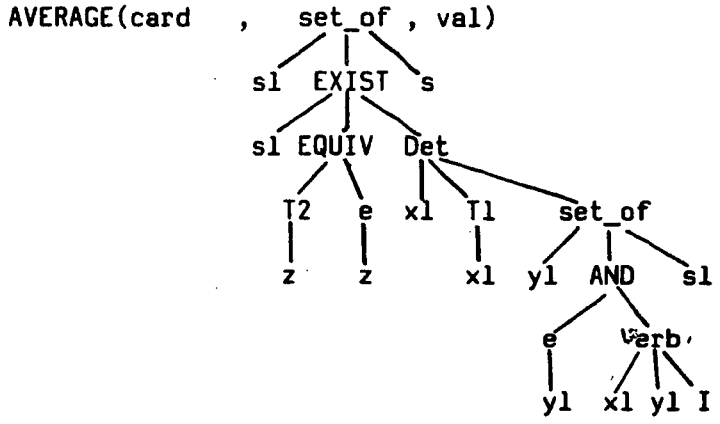
where T1 is the translation of the noun-phrase N1 and T2, that of N2, is rewritten into :



n5 is the number of elements y that satisfy the formula given above. An average value with either equivalences to T2 (rep. I6) or to T1 (rep. I7) is computed and compared to n5.

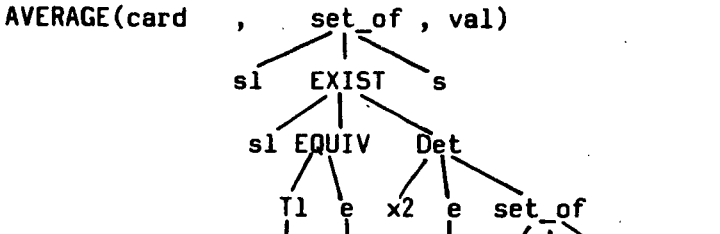
AVERAGE is defined as follows, depending on location of focus:

(1) with focus on N2:



Representation I6

(2) with focus on N1:



Representation I7

Notice that Det is translated by another rule into its final form, i.e. it is independent of the translation of many.

Some properties of these representations

First, it should be noted that scoping problems are assumed to be solved

before applying the rewrite rules described above, possibly with references to contextual knowledge [25].

The way representations are computed is modular since rewriting rules may be applied in any order on the input three branched quantified tree. Modularity is also a characteristics of the global representation of a determiner since fuzziness is encoded at the level of the comparison predicate, independently of the translation of the determiner whereas ambiguity is expressed via the specification of as many lexical entries as there exist relevant interpretations of a determiner.

Next, the representations we have defined remain valid for more complex structures than those presented in the above examples since no hypothesis is made about the nature of structures T and S. In particular, in T and S any noun may include pre- and post- modifiers. However, in this case, we will have to deal with focus applied on either the noun or one of its modifiers (cf. section 3).

Finally, the following array sums up the conditions of use of each representation:

	A	B	C
Many introduces a noun in the subject NP	I1 I2 I4	I3	I5
Many introduces a noun in a complement NP	I2 I4	I6	I7

A: no focus,

B: focus on the noun introduced by "many",

C: focus, but not on the noun introduced by "many", i.e. with focus on a complement for I5 and on the subject of the clause for I7.

Dealing with ambiguities

The above array that sums up the conditions of use of the different representations shows that only two cases are subject to ambiguity. Both cases appear when no focus is involved. Focus is, in fact, a very powerful mean to eliminate ambiguities if, of course, its determination is not ambiguous by itself.

Representation I4 is used either when "many" introduces a noun complement (in a subject or complement noun-phrase):

"The owners of many cars have to pay a tax. "

or when it introduces the head noun of a complement noun-phrase:

"Paul has invited many people to his party. "

Thus, elementary syntactic information about grammatical function is used to select I4. When "many" is in the subject NP, if I4 cannot be selected then I1 is the alternative.

Representation I2 seems to us to be rather devoted to specific cases. For instance, if we state:

"Chopin wrote many Mazurkas. "

without any focus either on Chopin or on Mazurkas, then no comparison criteria can be invoked. Only an "absolute" comparison, with a number fixed a priori, can be forseen.

A very simple criteria to decide whether I2 or I4 is the correct representation when many introduces the head noun of an NP complement is to select I2 only if I4 is "meaningless" or "absurd" in the sense given to these terms in presuppositional logic [4]. In other terms, I2 is selected if the

predicate AVERAGE in I4 fails, i.e. if no information can be found in the context to compute an average value. If no other composer than Chopin is known to have written Mazurkas, no average number of Mazurkas per composer can be computed.

3_ GENERALIZATIONS, EQUIVALENCES and FOCUS

In this section, we introduce a very simple way to deal with generalizations and equivalences. Although a lot of work remains to be done about that point, the method proposed seems to work for several well-defined applications.

GENERALIZATIONS AND REPRESENTATIONS WITH FOCUS

If we consider the sentence:

"Many French artists live in poor conditions. "

focus in the subject NP may be on either "french" or "artist". With focus on the intersective adjective "french", the sentence means that in comparison with the artists of the other countries known to the knowledge base, more French artists live in poor conditions. In this case, the generalization is made on the adjective. A very simple, but efficient, way to proceed consists in removing the adjective on which the focus is away from the noun phrase. This is realized by the 3 argument predicate GENERALIZE:

```
GENERALIZE(<formula>,
           <focus>,
           <generalization>)
```

For our previous example, we have:

```
GENERALIZE(AND(french(x),artist(x)),french(x),artist(x)).
```

This approach for the generalization function seems to work also for scalar adjectives and restrictive relative clauses [27]:

"Many famous symphonies by German composers are very often played by orchestras. "

with no focus on the subject NP, may lead to several generalizations, depending on focus, as, for instance:

GENERALIZE(famous symphonies by German composers,famous,symphonies by German composers),

GENERALIZE(famous symphonies by German composers,German,famous symphonies {by composers}).

(for more clarity, we have written the input sentence in the arguments of GENERALIZE rather than logical formulas).

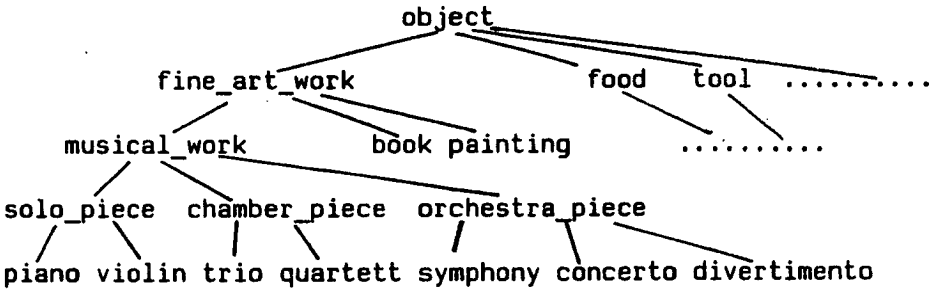
Notice that removing the adjective on which the focus is entails that adjectives which are not necessarily at the same level of generality (e.g. French and European) have the same generalization. Two more accurate but more complex and far less efficient approaches are possible. The first approach is to define a hierarchy between adjectives as we do below for nouns. However, this hierarchy depends, in most cases, on the noun to which the adjective is associated. This entails a very detailed level of specification of knowledge in the knowledge base. The second approach integrates considerations on focus, e.g. if a dialog is about Europe and if we state that "many French workers have a car", then French would probably have to be generalized into European. To realize this, it is necessary to define an hierarchical structure of focus and sub-focusses encountered in the dialog or in the text associated with their modifiers. This latter appraoch seems to us to be more computationally tractable than the first one.

Another crucial problem is to find appropriate generalizations to nouns, e.g. how "artist" generalizes into "worker" or "people" and "symphony" into "orchestra piece" or "musical work". With focus on the main noun "artist", the sentence "many French artists live in poor conditions" means, for

instance, that in comparison with other french people, more artists live in poor conditions. The generalization is made on the noun:

GENERALIZE(AND(french(x),artist(x)),artist(x),AND(french(x),people(x))).

When a knowledge base about a precise domain is built, a possible methodology to determine appropriate generalizations is to define hierarchies between concepts. A traditional approach to representing this hierarchy is to use an oriented acyclic graph:



This hierarchy, called IS-A hierarchy, is strongly domain-dependent, even if some common sense principles chair its construction. In PROLOG, it can be represented by facts :

is_a(symphony(X),orchestra_piece(X)).

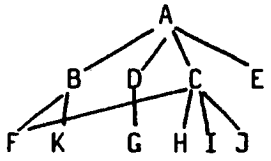
is_a(concerto(X),orchestra_piece(X)).

This type of hierarchy is also very often used in natural language systems as in [28] and [29] to deal with semantic feature checking.

The next point is to know at what level in the hierarchy is made the generalization. In fact, the level strongly depends on context and particularly on what the text or the conversation is focussed. Many works have been devoted to that point [12], thus, we assume that focus can be determined and that the generalization by default is made at the level immediately above the entity to generalize.

EQUIVALENCES AND REPRESENTATIONS WITH FOCUS

The notion of sister-concept introduces the necessity of defining an equivalence relation in the hierarchy defined above for concepts and entities. We say that the immediate sister-concepts of a concept C in an oriented acyclic graph are all the concepts which are the immediate daughters of the mother node of C. For instance, in the graph :



the immediate sister-concepts of H are F, I and J.

Next, we say that the level n sister-concept of a concept C are all the concepts which are the daughters at a distance n down in the graph from the node which is at a distance n up in the graph from C. For instance, the level 2 sister-concepts of H are K, F, G, I and J, the mother node being A, which is at a distance 2 up in the graph from H. Assuming that current focus is unique, there will have no ambiguity on the identity of a mother node.

Equivalences are determined by the following PROLOG predicate:

EQUIV(<concept>,
<level in the hierarchy>,
<list of sister-concepts>).

For the previous example, we have:

EQUIV(H,1,[F,I,J])

The level of hierarchy depends on the focus, in a way similar to the generalization function. Our approach entails that, in an is-a hierarchy of concepts, sister-concepts must be assigned the same degree of generality (or particularity).

The problem of finding appropriate equivalences can be very complex and

dependent not only on the concept introduced by "many", but also on other elements in the sentence and in the context. To give a flavour of the complexity of knowledge involved comsider the following (simple) sentences:

A1: "Chopin wrote many Mazurkas. "
A2: "Chopin wrote many Sonatas. "
B: "In this concert, the solist played many Mazurkas. "
c: "Toscanini recorded many operas. "

In A1 and A2, the same comparison criteria cannot be used because Mazurkas are much shorter than Sonatas. Thus, a hierarchical schema should be defined with a high precision, so as to allow comparisons only between pieces of approximately the same duration. In B, some kind of temporal reference should be considered, saying, for example, that 40% of the duration of the concert was devoted to Mazurkas, which makes a fair large number of pieces, compared to Sonatas. Finally, sentence C, with focus on "opera" entails a comparison with the number of other pieces recorded by Toscanini (symphonies, ...), but the computation should be made on the number of recorded pieces, allowing duplicates, because Toscanini has recorded several times some operas and other pieces at different stages of his life, which results in slightly different recordings.

It is not our purpose to study in detail the very structure of a knowledge base. In fact, much work remains to be done to establish a methodology to define real and efficient knowledge bases [14,22]. What we do here is rather to point out some features a knowledge base should have in order to allow adequate representations of determiners.

Ambiguities and cooperative responses

Hitherto, we have assumed that current focus can always be determined, and, futhermore, without any ambiguity. In this section, we assume that

focus cannot be determined or that its determination is not included in the current system. But, as focus is a reality, it is, however, necessary to take into account all the representations given here even if focus is not involved. This means that several representations are possible candidates to represent "many", either in a subject or in a complement NP.

This state of affairs entails that the system, that cannot use knowledge to select the correct representation, has to solicit the user via a request throwing ambiguities into relief. The problem is that it is often very complex, at a linguistic level as well as at logical and cognitive levels, to produce paraphrases to such ambiguities. Futhermore, it is not obvious that the user will be either able or willing to understand them.

Our approach is to give to the user a cooperative response [11,32] that describes the conditions under which this response is correct. We do not wish to go futher since this point is still under study, but the following example illustrates our approach:

User: *"Do many French artists live in poor conditions ?"*

System: *"If I compare to other artists in the world, many French artists do not live in poor conditions; on the contrary, if I compare to French workers, many French artists live in poor conditions. "*

Cooperativity can be more elaborated by giving, for instance, the percentage of artists that corresponds to the user's request.

It appears that the respresentations we have given can be used quite straightforwardly to generate parts of cooperative responses. They are viewed, in this case, as justifications of an answer to a request.

4_ ON REPRESENTING OTHER CONTEXT-DEPENDENT DETERMINERS

The seven representations given above for "many" form, in fact, a general framework which can also be used for representing other context-dependent determiners. We briefly consider in this section the most usual ones.

We view **few** as the symetric of many and, thus, the representations given above can be used in the same conditions. The PROLOG predicate LE (Less than or Equal) is used instead of GE. However, an additional constraint should be added that states that the sets which are built are never empty. Otherwise, this would entail that "no" implies "few". This additional constraint is implicit if the PROLOG call set_of(x,P(x),s) fails if there is no x satisfying P(x).

The determiner **most** does not allow as many distinct representations than many. Only representation I1 can be used, but with two meanings:

(1) Most can be equivalent to "more than half". I1 is used with comparison predicate GE(n,m(n1)) instanciated to : GE(n,0.5*n1).

(2) Most can also be roughly equivalent to "almost" [34,35]. In this case, GE(n,m(n1)) is instanciated to GE(n,0.9*n1) or some value around 0.9.

Several and **some** are represented by either I1 or I2. I3 is not a possible representation since there is no implicit comparison with set of entities other than the set of elements that satisfy T(x) and ¬S. Focus seems not to be meaningful and discriminatory for these determiners.

Finally, **much** and **little** are represented in the same spirit than many and few. However, much and little most of the time apply to non-enumerable quantities, thus, cardinality cannot be used. In fact, depending on the noun

they introduce and, to some extent, depending on context, a measure such as weight, volume, speed or intensity can be associated to these determiners. Then, average weight, volume or speed can be computed as previously and comparison predicates like COMP in [27] can be used. Important similitudes with the representation of scalar adjectives [27] should be noted.

In the sentences:

"There is much wind to day. "

"Some animals need only little food every day. "

"Much effort was devoted to understanding hieroglyphics. "

the measure units associated to "much" or "little" could be, respectively, the speed, the weight and the length of work in years or number of people. These measure units are, in fact, related to scalar properties of the noun introduced by the determiner.

The determination of the adequate property (and thus measure unit) is done at the level of the knowledge representation system. When concepts in sentences are translated into their domain-dependent representations, as in [22], they can quite directly be associated with facts of the knowledge base. For instance, facts concerned with what is stated in the first and last sentences above could be of the form:

```
wind(<identifier>,
     <speed in km/hour>,
     <degree of humidity, in percentage>,
     <direction>,
     <date>).
for instance:  wind(X,75,80,West,1.03).

and:
research_activity(<name of researcher>,
                  <length of work in years>,
                  <branch of research>).
for instance:  research_activity(champolion,45,egyptology).
```

The selection of an appropriate measure for a determiner seems to be deductible from the knowledge of the meaning of arguments of such facts. In very simple and limited cases, this selection can be done directly from the fact, without the need of domain specific knowledge. However, quite often, additional domain-dependent information is necessary to deduce, for instance, that "much wind" is equivalent to " a high speeded wind" and that "much effort" means "many years of work". Futhermore, other elements in the sentence, such as adjectives and relative clauses, may influence this selection.

Finally, adverbs of degree can modify the kind of determiners we study in this work: *very much, rather little, ...* Adverbs applied to determiners are represented in a way exactly similar than when they modify scalar adjectives [27]. For "many" and "few", the property involved is cardinality.

5_ TRUTH PERSISTENCE OF A REPRESENTATION

In this section, we study the persistence of the truth value of the representation of a determiner, depending on the evolution of the knowledge base, characterized by adjunction or deletion of facts or rules. This allows one to better control the truth value of a formula depending on the evolution of the knowledge base when this formula is interpreted either as a fact, a rule or as an integrity constraint. This leads us to define a notion of continuity and monotonicity for semantic representations with context-dependent determiners. It is necessary to do this study here since there are no general results in logic about that point when cardinality is involved.

First, it should be noted that the determiner representations we propose are independent of the domain of application in the sense that they do not

involve any particular basic entity, fact or property. Thus, they inherit of a well-known result of the model-theoretic approach: a determiner representation and, consequently, its truth value is insensitive to permutations of basic entities in a given knowledge base and, more generally, to bijections between knowledge bases. These properties are called respectively closure under permutation and closure under isomorphism [34].

Evolution of the knowledge base

Reasoning systems in Logic Programming usually use facts, rules and integrity constraints to build computational models of knowledge. To keep these models consistent with sequences of deletions and additions of information in the knowledge base, the reasoning programs frequently have to update parts of their models. The reasoner makes changes in the current models by desinvalidating some of the current assumptions and by adding some new ones. The usual strategy is to desinvalidate the smallest set of facts and rules to restore consistency. Valid facts and rules are marked "in", while the others are marked "out":

info(<logical representation of the statement>,<in/out>).

A fact or a rule remains "out" until the reasoner specifically puts it back into the current set of valid statements. Our approach is, thus, very close in its spirit to Truth Maintenance Systems [10].

In this paragraph, we study the characterization of the persistence of the truth value of a representation depending on the evolution of ground data in the knowledge base. More precisely, we study elementary evolutions of a knowledge base (a fact is added or deleted) and give the conditions that guarantee the truth persistence of a statement. If these conditions are not met, then the truth value of the statement may be affected and, thus,

controls have to be done. This study is restricted to the representations developed in this work. It follows two kinds of results for sentences interpreted either (1) as facts or rules or (2) as integrity constraints.

From the point of view of facts and rules, this would enable the system, at any time, to indicate if a given statement remains valid or risks to be no longer valid. From the point of view of integrity constraints, this would have rather to be considered as giving restrictions on the evolution of data in a knowledge base in order to guarantee the monotonocity of a set of statements interpreted as integrity constraints. Truth persistence is indeed a main characteristics of integrity constraints.

A crucial problem is non-atomicity of statements: each rule cannot be viewed as an isolated element. Due to the presence of several rules in the knowledge base, other facts than those explicitly affected by the update could change. It is necessary to consider the extension of other rules each time something is changed in the database. However, we can strongly reduce the difficulty of integrity constraint checking by using the results of theorem 3 given in [16], which gives an insightful simplification when updating a database. A set, called Θ, describes all the ways the addition of a clause C to a database may affect a J-instance (J is a preinterpretation) of a negated atom in a formula W.

In this subsection, we characterize the cases for which Θ (more precisely, $\Theta \cup \Psi$) is empty and, thus, for which the truth value of the corresponding fact, rule or integrity constraint is not affected. We will go further in the next subsection by introducing continuity and monotonicity.

We now summarize our main results (which can be demonstrated straighforwardly). We consider that T and S in: det(x,T(x),S) can be

associated to the finite set of elements that satisfy them. Then, we have the following consequences on the evolution of either T or S. For representations I1 to I5, we distinguish three relevant cases:

- (D) the set of elements that satisfy S becomes greater (GR) or smaller (SM), the set of elements that satisfy T remains unchanged but $T \cap S$ may change.
- (E) the set of elements that satisfy T becomes greater (GR) or smaller (SM) and the set of elements that satisfy S remains unchanged: this entails that $T \cap S$ cannot change.
- (F) same conditions than in (D) but $T \cap S$ remains unchanged.

Then, we have the following results for I1 to I5:

(A "-" means that the truth value of the logical representation may be affected; a "+" means that it is not.)

		D	E	F
SM	I1	-	+	+
	I2	-	+	+
	I3	-	+	+
	I4	-	+	+
	I5	-	+	+
GR	I1	+[2]	-	+[3]
	I2	+	+	+
	I3	-	-	-
	I4	+	-	+
	I5	+[4]	-	+[4]

[2] For instance, if we have the sentence:

"Many workers have a car. " and if the property of having a car is added to some entity in the knowledge base, the truth value of the sentence is not affected if "many" is represented using I1.

[3] In this case, the information added to the knowledge base is not about a worker. Thus, the truth value of the sentence with representation I1 is not affected.

[4] We have "+" only if no information is added about sister-concepts that may satisfy S.

Similarly, the truth persistence for representations I6 and I7 can be caracterized. The original form of the input structure being:

Det(x,T1(x),Many(y,T2(y),Verb(x,y,I)))

we distinguish four cases:

- D, E and F as above, with : S= T2(y)∧Verb(x,y,I) and T=T1.
- G: T and S remain unchanged but the set of elements that satisfy the formulas:
  - •(G1): T2'(y)∧Verb(x,y,I) where T2' is a sister-concept of T2 has increased (GR) or decreased (SM) (cf. I6),
  - •(G2): T1'(x)∧T2(y)∧Verb(x,y,I) where T1' is a sister-concept of T1 has decreased or increased (cf. I7).

Then, we have the following results:

		D	E	F	G
SM:	I6	-	+	+	+(G1)
	I7	-	+	+	+(G2)
GR:	I6	+	+	+	-(G1)
	I7	-	+	-	-(G2)

The results above concern the adjunction of either ground facts or rules. In the case of adjunction of a rule, it is necessary, roughly speaking, to consider the extension associated to this rule in the current

knowledge base.

Continuity and Monotonicity

We now generalize and formalize the previous results. Results have already been stated for determiners which are first-order definable within the model-theoretic framework [30,34,35], where monotonicity and continuity have been demonstrated. Our approach is more complex since our representations are not strictly first-order ones due to the presence of aggregating functions (e.g. set_of, card, AVERAGE, ...). We show that continuity holds and allows additional simplifications for controlling the truth value of facts and rules and for checking integrity constraints. However, monotonicity, in general, does not hold.

We first define continuity on T and S separately. Let $R(x,T(x),S(x,...))$ be the general form of a determiner representation, then, we define: X, X' and X" as being the sets of the elements x that satisfy T(x) in three different states I, I' and I" of the knowledge base. In the same way, let Y, Y' and Y" be the sets of all the elements x that satisfy S(x,....) in I, I' and I". Then, we have the following definitions:

For more clarity in what follows, we directly write T, T' and T" to represent T in state I, T' in state I' and T" in state I". T in the states I, I' and I". The same is done for S.

Continuity on T:

$(\forall X, X', X'')\ ((X \subseteq X' \subseteq X'') \wedge (X \cap Y = X' \cap Y = X'' \cap Y) \wedge R(x,T,S) \wedge R(x,T'',S)) \Rightarrow R(x,T',S).$

Continuity on S:

$(\forall Y, Y', Y'')\ ((Y \subseteq Y' \subseteq Y'') \wedge (X \cap Y = X \cap Y' = X \cap Y'') \wedge R(x,T,S) \wedge R(x,T,S'')) \Rightarrow R(x,T,S').$

Both continuity on T and S work for representations I1 to I7.

Demonstrations follow quite straightforwardly and are all built on the same schema. We give here the demonstration for Il and continuity on T:

At the state I, we have the formula :

EXIST(sl,set-of(xl,T(xl),sl),EXIST(s,set-of(x,AND(T(x),S),s),

AND(card(s,n),and(card(sl,nl),F(n,m(nl))))))).

and at the state I" :

EXIST(sl",set-of(xl,T(xl),sl"),EXIST(s",set-of(x,AND(T(x),S),s"),

AND(card(s",n"),AND(card(sl",nl"),F(n",m(nl"))))))).

In addition, we have the hypothesis:

$X \subseteq X' \subseteq X''$ and $X \cap Y = X' \cap Y = X'' \cap Y$

from which we can deduce:

(1) $n \leq n' \leq n''$ and (2) nl = nl' =nl".

The implication to demonstrate depends here in fact on the truth value of the comparison predicate, thus, we have to show that:

For any $F \in \{LE, EQ, GE\}$, $F(n,m(nl)) \wedge F(n'',m(nl'')) \Rightarrow F(n',m(nl''))$

from (2) we deduce:

$F(n,m(nl')) \wedge F(n'',m(nl')) \Rightarrow F(n',m(nl'))$

then, from (1) it is clear that if both n and n" satisfy F then n' satisfies it also. Then, we have the result:

R(x,T'(x),S(x,...)) is true.

From continuity on T and S we can deduce by recurrence the property of continuity:

$(\forall X, X', X''), (\forall Y, Y', Y''), ((X \subseteq X' \subseteq X'') \wedge (Y \subseteq Y' \subseteq Y'') \wedge (X \cap Y = X' \cap Y' = X'' \cap Y'') \wedge$

$R(x,T,S) \wedge R(x,T'',S'')) \Rightarrow R(x,T',S')$.

These properties are very useful to control the validity of facts and rules

and to check integrity constraints. If several ground facts of the same kind are deleted or added to the knowledge base, then it suffices to consider the initial and the final state of the updating process provided that $X' \cap Y$ remains unchanged. If continuity does not hold, then, for integrity constraints, intermediate stages have to be considered. These properties associated to theorem 3 of [16] yield to a more efficient and more acceptable system.

An extension of continuity is monotonicity, which could be defined as follows within our framework:

$(\forall X, X'), (\forall Y, Y') ((X \cap Y = X' \cap Y') \wedge (X \subseteq X') \wedge R(x,T,S) \Rightarrow R(x,T',S')$.

This property does not hold in general for the representations defined here.

6_ CONTEXT-DEPENDENT DETERMINERS AND DEFAULT REASONING

Default reasoning has become very commonly used in AI systems [24,2]. Roughly speaking, default reasoning is based on the following inference schema :

"In the absence (in the current knowledge base) of any information to the contrary, then deduce .... "

There exist different forms of defaults; we will be concerned here with normal defaults, which are of the form:

$$\frac{P(x) : R(x)}{R(x)}$$

This notation can be paraphrased by:

"Given x and the prerequisite P(x), if there is no contradiction to R(x) then infer R(x). "

Default reasoning seems to be an appropriate inference schema to treat requests whose evaluation cannot be done due to a lack of information. For

several kinds of statements containing determiners such as "many" or "few" and interpreted as facts, rules, it seems to us to be more relevant to interpret them as default rules rather than classical rules. This is mainly due to the fact that "many" or "few" do not introduce universal quantification. This point is exemplified in dialog 1 in the introduction. As a second example, consider the statement:

"Many workers travel by train. "

which rather entail the production of the following default rule:

$$\frac{worker(x):\ (\exists y,\ train(y) \wedge to\_travel\_by(x,y))}{(\exists y,\ train(y) \wedge to\_travel\_by(x,y))}$$

However, the sentence above can be interpreted as a fact or a rule, which, thus, may be valid or not (i.e. "in" or "out") depending on updatings of the knowledge base, as explained in section 5:

info(Many(x,worker(x),EXIST(y,train(y),to_travel_by(x,y))),<in or out>)

(for more clarity, we have not translated many in its final form).

Given this rule, a more accurate representation of the default would be:

$$\frac{worker(x) \wedge rule(many(x1,worker(x1),EXIST(..)),in)\ :(\exists y,train(y) \wedge to\_travel\_by(x,y))}{(\exists y,train(y) \wedge to\_travel\_by(x,y))}$$

It seems to us that adding the constraint that the rule corresponding to the default is valid (i.e. "in") in the prerequisite of a default rule guarantees that the inference by default is made only when "many x S(x,...)" is satisfied. To our opinion, this approach gives a stronger and more accurate semantics to default rules.

Indeed, we feel that a default without this constraint on the validity of the corresponding fact or rule cannot really represent "many" or "most". We think that such a default should rather be interpreted as a default

option, as those used in frames and scripts to fill slots.

Another point is that is seems to us that a default rule that really means "many" or "most" should be a powerful mean to restore consistency in knowledge bases whereas ordinary defaults should not because they have a much less stronger meaning.

Finally, it should be noted that if a statement with a context-dependent determiner is interpreted as an integrity constraint and if it can be represented by a default then the constraint on the validity of the corresponding statement is always true since it is an integrity constraint.

We now show how a default can automatically be built from the semantic representation of utterances containing "many" or "few". Not all sentences are at the origin of the production of a default. It seems to us that depending on syntactic and semantic factors, we have three situations:

(1) A sentence has to be represented as a default rule and not as a classical rule. This is the case for sentences where "many" introduces the head noun of the subject noun-phrase.

(2) A sentence can both be represented by a default and by a classical rule, fact or integrity constraint. Then, depending on the kind of request that will appear, either the default or the classical rule will be used. Sentences with "many" introducing the head noun of an object noun-phrase and with the subject noun-phrase universally quantified fall in this class.

(3) Finally, other sentences, and in particular those where "many" introduces a noun which is a modifier of another noun cannot be represented as a default.

The nature of the representation (a default or a classical representation) is

selected depending on:

- the kind of representation used for many (cf. the array at the end of section 2 that sums up the conditions of use of each representation)
- the quantification on the subject noun-phrase for the case (2).

We now illustrate the different situations and give the way default rules are produced.

We first consider representations of sentences with "many" dominating all the other determiners of the sentence. This is the case, for instance, of the above example. Let the semantic representation of such a sentence be of the form:

Rm(x,T(x),S(x,...))

then, we have the following rule:

rule(Rm(x,T(x),S(x,...)),<in or out>).

and the default:

$$\frac{T(x) \wedge rule(Rm(x1,T(x1),S(x1,...)),in) : S(x,...)}{S(x,...)}$$

Sentences in the situation (2) stated above also entail the production of defaults. More formally, sentences with a semantic representation of the form:

U(x,T1(x),Rm(y,T2(y),S(y,...))) (U is the representation of "all the")

entail the production of the default:

$$\frac{T(x) \wedge rule(U(x1,T(x1),Rm(y1,T2(y1),S(y1,...)),in): S(x,...)}{S(x,...)}$$

However, the evaluation of a subsequent request is made using either the

default or the classical representation depending on the kind of request. For instance, if the following statement is in the state "in":

"All the musicians know many musical works. "

and if we ask:

"Does Edith know Petrouchka ? "

knowing:

musician(edith).

musical_work(petrouchka).

and if nothing is in contradiction with:

to_know(edith,petrouchka).

then we can deduce that Edith knows Petrouchka using the default rule.

On the contrary, if we say:

"Does Kate know many musical works ?"

knowing that Kate is a musician, then a positive answer is produced from the classical rule. It seems to us that when the request is about a precise instance of the entities quantified by "many" then the default should be used. In the other cases, the classical rule should be prefered.

Finally, sentences containing a determiner such as "few" entail the production of defaults with a negation on S(x,...), or at least a negation on the verbal predicate, depending on what is presupposed [27]. For instance, if we have the following statement:

"Few people travel by plane. "

which is in the state "in", and if we ask:

"Has Edith travelled by plane ? "

in case of a lack of information about Edith's trips by plane, we would deduce that she has not travelled by plane. Let such sentences be of the

form:

Rf(x,T(x),S(x,...)) where Rf is any representation of determiners like "few".

The corresponding rule is:

rule(Rf(x,T(x),S(x,...)),<in or out>).

and the default is:

$$\frac{T(x) \wedge rule(Rf(x1,T(x1),S(x1,...)),in): \neg S(x,...)}{\neg S(x,...)}$$

7_ INPLEMENTATION

The output of the syntactic parsing process is a three branched quantified tree with an additional argument added to nominal, adjectival and verbal predicates, as described in section 2. The tree is then rewritten during the semantic process, using the rules given in section 3. This rewriting process is used in the same manner for adjective phrases and adverbs [27].

The implementation of the rewrite rules has been carried out and is quite elementary; thus we only give here the main lines of the overall architecture. First, as the representations proposed here can be used for several determiners, we have defined prototypes with parameters; these are, for instance, the prototypes associated to I1, I2 and I3:

```
determiner_sem_form(select(subj,head,nofocus),
                    *x,*F,*coef,*T,*S,exist(*s1,set_of(*x1,*T1,*s1).
                    exist(*s,set_of(*x,*T.*S,*s),card(*s,*n).card(*s1.*n1).
                    MULT(*n,*coef,*nn).*F(*nn,*n1))))
   -other_var(*x.*T,*x1.*T1).

determiner_sem_form(select(*1,*2,nofocus),
                    *x,*F,*coef,*T,*S,exist(*s,set_of(*x,*T.*S,*s).
                    card(*s,*n).*F(*n,*coef))).

determiner_sem_form(select(subj,head,focuscpt),
                    *x,*F,*coef,*T,*S,exist(*s1,set_of(*x1,*T1,*s1).
                    exist(*s2,set_of(*x,*T1.*S,*s2).card(*s1,*n1).card(*s2,*n2),
                    exist(*s3,set_of(*x3,*GT3,*s3),
```

```
                    exist(*s4,set_of(*x4,*GT4.*S4,*s4),
                    card(*s3,*n3).card(*s4,*n4).DIV(*n1,*n2,*nn1).
                    DIV(*n3,*n4,*nn3).MULT(*nn3,*coef,*nn4).
                    *F(*nn1,*nn4))))))
   -other_var(*x.*T,*x1.*T1)
   -GENERALIZE(*x.*T,*x.*GT,*S)
   -other_var(*x.*GT,*x3.*GT3)
   -other_var(*x.*GT,*x4.*GT4).
```

*F refers to the comparison predicate and *coef is the arithmetical function. In this version of PROLOG, variables identifiers are preceded by a *, variables may appear as heads of predicates and the point is used here as an implicit "and" in the semantic representations. Representations I4 to I7 are represented in the same way. The first argument, called "select", is used to characterize the conditions of use of each representation (grammatical function, status and focus) as described in section 2.

We have only implemented very simple treatments for GENERALIZE. The call "other_var(*x.*T,*x1.*T1)" creates a new instance T1 of the sutree T where the variable *x becomes an new variable *x1. Its implementation is not very elegant in PROLOG:

```
other_var(*x.*T,*x1.*T1)
   -assert(fact(*x,*T))
   -fact(*x1,*T1)
   -delete_clause(fact).
```

A more general and built-in PROLOG call would be desirable to create instances of given subtrees where any number of variables could be renamed upon request.

The general call of the rewriting process is realized by the call "rel" as follows:

```
rel(*N(*X,*T,*S),*NN)
   -//
   -rewrite_restr(*T,*T1,*Gf,*St,*Foc)
   -rel(*S,*S1)
```

```
   -rewrite_node(*N(*X,*T1,*S1),*NN,*Gf1,*St1,*Foc1).
rel(*Verb(*grf,*1,*2),*Verb(*1,*2)).

rewrite_restr(and(*T1,*T2),and(*TT1,*TT2),*Gf,*St,*Foc)
   -//
   -rewrite_restr(*T1,*TT1,*Gf,*St,*Foc)
   -rel(*T2,*TT2).
rewrite_restr(*T(noun(gr(*Gf,*St,*Foc)),*X),*T(*X),*Gf,*St,*Foc)
   -focus(*T,*Foc).

rewrite_node(many(*X,*T,*S),*Z,*Gf,*St,*Foc)
   -determiner_sem_form(select(*Gf,*St,*Foc),*X,ge,2,*T,*S,*Z).
```

If several levels of the input tree are concerned by the rewriting (cf. I4, I6, I7) then the first argument of "rel" is decomposed into the necessary number of levels. In fact, we have two calls "rel", with either a one or a two level decomposition of the input tree.

Finally, to some prototypes are attached preconditions of selection, as explained in section 2. For instance, to I2 is attached the restriction that it is selected only if I4 is meaningless, i.e. that AVERAGE in I4 fails. For that purpose, we add to the prototype of I2 the restriction:

```
   -meaningless(AVERAGE(card,set_of(*s1,exist(*x1,*T1,
      set_of(*y1,exist(*y,*T2.*TT1,*s1),*s2,*val))).
(T1, TT1 and T2 are defined as in I4.)
```

There are several ways, more or less efficient, to compute "meaningless" that we shall not detail here. For instance, it suffices to prove NOT(*T2) or NOT(*TT1).

The production of a default rule is implemented straightforwardly from the final semantic representation of sentences interpreted as rules. We have not yet implemented the truth persistence system because we wish to implement a truth persistence system that is adapted to various structures which are still under study (e.g. adverbs, adjective phrases, ...).

The overall performances of the rewriting process are good, compared to the syntactic analysis process, due, in fact, to a higher level of determinism. However, terms become quickly very large and some optimizations would be desirable to have an efficient evaluation process.

CONCLUSION AND DISCUSSION

In this work, we have proposed several semantic representations for context-dependent determiners such as "many" or "little", with a degree of precision we think to be adequate for man-machine dialogs. These representations introduce some requirements on the structure of the knowledge base in order to allow the computation of generalizations and equivalences of concepts. In addition, focus and grammatical functions have been shown to be of a crucial importance to deal with ambiguities. Finally, we have introduced some ideas and properties about the truth persistence of formulas containing such determiners and interpreted either as rules or integrity constraints when the knowledge base is updated. We have also shown that several sentences should rather be represented by default rules than by traditional rules or facts. A method to build default rules has been introduced.

We feel that this work also introduces a methodology in the study of the semantic representation of entities and structures in logic programming. For instance, we think that other syntactic (sub)categories, such as adverbs should be represented in a way quite similar, with constraints of the same kind on the knowledge base. This has already been pointed out to some extend for quantificational adverbs (seldom, often, ...) in [19,20]. Structures like comparatives and superlatives should also be concerned with problems of truth persistence and default reasoning with a quite similar approach.

Another point is to extend our present formalism to sentences where structures directly involved in the construction of the semantic representation of a determiner are not noun phrases but, for instance, adverbs, as in:

"Many women work outside nowadays. "

where focus is on the temporal adverb "nowadays". A comparison with previous years (or other periods) is necessary. However, in such cases, the process of finding equivalent concepts has to be extended.

Finally, it should be noticed that the representations we propose here are not, in general, final logical formula, ready to be evaluated on a knowledge base [22,33]. The main reasons are that some elements may require a deeper understanding where non-local references and metaphors are solved. Secondly, transformations of subtrees into more appropriate ones linked to the undelying application have often to be done. Transformations may also be done for efficiency reasons. These transformations may not be really linguistically motivated. Finally, the generality and the explanatory power of the model, even at a structural level, may be lost by ad'hoc transformations.

REFERENCES

[1] Abramson, H., Definite Clause Translation Grammars, *Proc. of the intern. symposium on Logic Programming,* Atlantic City, NY, 1984.

[2] Besnard, P., *Introduction to Default Logic,* to appear end of 1986, Springer Verlag.

[3] Colmerauer, A., An Interesting subset of Natural Language. in: *Logic Programming,* CLARK and TARNLUND Edts. Academic Press. 1982.

[4] Colmerauer, A., Pique, J.F., About Natural Logic, *Advances in Database Theory* Vol. 1, Plenum Press, 1980.

[5] Dahl, V., Un système déductif d'interrogation de banques de données en espagnol, Thèse, Université de Marseille-Luminy, GIA, 1977.

[6] Dahl, V., Quantification in a three-valued logic for natural language question-answering systems, *6th IJCAI Tokyo,* 1979.

[7] Dahl, V., Translating Spanish into logic through logic, *Computational Linguistics,* Vol. 7-3, 1981.

[8] Dahl, V., Abramson, H., On Gapping Grammars, *Proc of the 2nd logic programming conference,* Uppsala university, 1984.

[9] Dahl, V., More on Gapping Grammars, *Conference. on Fifth Generation Computer systems,* Tokyo, 1984.

[10] Doyle, J., A Truth Maintenance System, *Artificial Intelligence,* Vol. 12, pp.231-272, 1979.

[11] Gal, A., Minker, J., A natural Language Interface that Provides Cooperative Answers, *Proceedings of the second conference on artificial intelligence applications,* Miami, 1985.

[12] Grosz, B., The representation and Use of Focus in a System for

Understanding Dialogs, *Proceedings of 5th IJCAI*, 1977.

[13] Gallaire, H., Minker, J., *Logic and databases: a deductive approach,* Computing Survey, 1983.

[14] Hobbs, J., Building a Large Knowledge Base for a Natural Language System, *Proceedings of COLING-84*, 1984.

[15] Lloyd, J.W., *Foundations Of Logic Programming,* Springer Verlag, 1984.

[16] Lloyd, J.W., Topor, R.W., A Basis for Deductive Database Systems, *Journal of Logic Programming*, pp. 93-109, 1985.

[17] Lloyd, J.W., Topor, R.W., A Basis for Deductive Database Systems II, *Journal of Logic Programming*, pp. 55-67, 1986.

[18] McCord, M., Using Slots and Modifiers in Logic Grammars for Natural Language, Technical report 69-80 (1980), and *Artificial Intelligence,* 1982.

[19] McCord, M., Focalizers, the scoping problem and semantic interpretation rules in logic grammars, Tech. report Univ. of Kentucky (1981), also in: *Logic programming and its Applications*, D. Warren and M. van Caneghem Edts, Ablex, 1986.

[20] McCord, M., Semantic Interpretation for the EPISTLE system, *Proceedings of the second international conf. on Logic Programming* , Uppasala University, 1984.

[21] McCord, M., Modular logic grammars, *Proceedings of 23rd ACL Meeting*, Chicago, 1985.

[22] Palmer, M., Driving semantics for a limited domain, Ph. D. Thesis, Univ. of Edinburgh, 1984.

[23] Pereira, F., Logic for natural language analysis, SRI international technical note no 275, 1983.

[24] Reiter, R., On Reasoning by Default, *Proceedings of TINLAP-2*, 1978

[25] Saint-Dizier, P., Handling quantifier scoping ambiguities in a semantic representation of natural language sentences, in: *Natural language Understanding and Logic Programming* V. Dahl and P. Saint-Dizier Edts, North Holland, 1985.

[26] Saint-Dizier, P., Expression of Syntactic and Semantic Features in Logic Based Grammars, *Computational Intelligence Journal*, Vol. 1, nb 3-4, 1986.

[27] Saint-Dizier, P., An approach to Natural Language Semantics in Logic Programming, *Journal of Logic Programming*, to appear end of 1986.

[28] Shieber, S.H., Uszkoreit, H., Pereira, F., Robinson, J., Tyson, M., The Formalism and Implementation of PATR-II, SRI report 1894, 1983.

[29] Schwind, C., Logic Based Natural Language Processing, in: *Natural Language Understanding and Logic Programming*, V. Dahl and P. Saint-Dizier Edts, pp. 207-219, North-Holland, 1985.

[30] Van Betlem, J., Questions about Quantifiers, *Journal of Symbolic Logic*, 1985.

[31] Warren, D.H., Efficient processing of interactive relational database queries expressed in PROLOG, *Proceedings of VLDB-81*, 1981.

[32] Webber, B.L., Question, Answer and Responses: Interacting with Knowledge Base Systems, Research Report, MS-CIS-85-50, University of Pennsylvania, 1985.

[33] Weischedel, R.M., Mapping Between Semantic Representations Using Horn Clauses, *Proceedings of Coling-84*, 1984.

[34] Westerstahl, D., Some Results on Quantifiers, *Notre Dame Journal of Formal Logic*, vol. 25 nb.2, pp. 152-170, 1984.

[35] Westerstahl, D., Logical Constants in Quantifier Languages, *Linguistics and Philosophy*, vol. 8, pp. 387-413, 1985.

[36] Zadeh, L.A., Test-score semantics for natural languages and meaning-representations via PRUF, Technical Note 247, AI Center, SRI International, 1981.

PI 286 — **La tolérance aux fautes dans un système temps-réel à contraintes strictes**
Maryline Silly – 32 pages ; Février 86.

PI 287 — **A new statistical approach for the automatic segmentation of continuous speech signals**
Régine André-Obrecht – 38 pages ; Mars 86.

PI 288 — **Synthèse sur les réseaux locaux temps-réel**
Philippe Belmans – 40 pages ; Mars 86.

PI 289 — **Calcul distribué d'un extrémum et du routage associé dans un réseau quelconque**
Jean-Michel Hélary, Aomar Maddi, Michel Raynal – 36 pages ; Mars 86.

PI 290 — **A new matrix multiplication systolic array**
Patrice Quinton, Brigitte Joinnault, Pierrick Gachet – 12 pages ; Avril 86.

PI 291 — **Une introduction à quelques techniques du contrôle distribué à travers un exemple**
Noël Plouzeau, Michel Raynal, Jean-Pierre Verjus – 22 pages ; Avril 86.

PI 292 — **Distributed Synchronization of Parallel Programs : why and how ?**
Patrice Quinton, Jean-Pierre Verjus – 16 pages ; Avril 86.

PI 293 — **Sur l'utilisation des séquences multi-images en robotique**
Lionel Marcé – 16 pages ; Avril 86.

PI 294 — **Stabilité de l'analyse factorielle des correspondances multiples en cas de données manquantes et de modalités à faibles effectifs**
Habib Bénali – 16 pages ; Avril 86.

PI 295 — **The problem of diagnosis with respect to physical parameters for changes in structures**
Georges V. Moustakides – 14 pages ; Avril 86.

PI 296 — **An overview of local environment sensing in robotics applications**
Bernard Espiau – 38 pages ; Mai 86.

PI 297 — **Context-dependent determiners in logic programming : semantic representations and properties**
Patrick Saint-Dizier – 54 pages ; Mai 86.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique